CLAIMS

What is claimed is:

1. A method for translating Verilog into C++, the method comprising:

searching for a Verilog pattern in a Verilog file, the Verilog pattern associated with a specific functionality; and

substituting the Verilog pattern with a C++ language expression, wherein the C++ language expression is associated with the specific functionality.

2. The method of claim 1, wherein the translating from Verilog into C++ utilizes macro functions in VBScript.

3. The method of claim 2, further comprising identifying whether the Verilog file comprises at least one of a task library, a main driver, and a driver module.

4. The method of claim 3, further comprising:

if the Verilog file comprises a task library:

identifying a Verilog task within the task library; and

translating the Verilog task into a C++ function; and

if the Verilog file comprises a main driver:

inserting in the Verilog file at least one C++ interface header.

5. The method of claim 1, wherein the Verilog pattern comprises # delay statements from the Verilog file.

6. The method of claim 1, wherein the Verilog pattern comprises `ifdef statements in the Verilog file.

7. The method of claim 1, wherein the Verilog pattern comprises ` symbols from the Verilog file.

8. The method of claim 1, wherein the Verilog pattern comprises a begin keyword in the Verilog file to a "{" symbol.

9. The method of claim 1, wherein the Verilog pattern comprises an end keyword in the Verilog file to a "}" symbol.

10. The method of claim 1, wherein the Verilog pattern comprises at least one register definition in the Verilog file into at least one C++ definition.

11. The method of claim 1, wherein the Verilog pattern comprises at least one combinatorial assignment in the Verilog file.

12. The method of claim 1, wherein the Verilog pattern comprises at least one event in the Verilog file into at least one C++ event.

13. The method of claim 1, wherein the Verilog pattern comprises at least one Verilog switch in the Verilog file into at least one C++ switch.

14. The method of claim 1, wherein the Verilog pattern comprises at least one Verilog concat expressions in the Verilog file into at least one C++ concat expressions.

15. The method of claim 1, wherein the Verilog pattern comprises at least one Verilog parameter in the Verilog file into at least one C++ #define.

16. The method of claim 1, wherein the Verilog pattern comprises at least one Verilog const in the Verilog file into at least one C++ const.

17. The method of claim 1, wherein the Verilog pattern comprises at least one Verilog bit access macro in the Verilog file into at least one C++ functional equivalent.

18. A machine-readable storage having stored thereon, a computer program having at least one code section for translating Verilog to C++, at least one code section being executable by a machine for causing the machine to perform steps comprising:

searching for a Verilog pattern in a Verilog file, the Verilog pattern associated with a specific functionality; and

substituting the Verilog pattern with a C++ language expression, wherein the C++ language expression is associated with the specific functionality.

19. The machine-readable storage according to claim 18, wherein the translating from Verilog into C++ utilizes macro functions in VBScript.

20. The machine-readable storage according to claim 19, further comprising code for identifying whether the Verilog file comprises at least one of a task library, a main driver, and a driver module.

21. The machine-readable storage according to claim 20, further comprising:

if the Verilog file comprises a task library:

code for identifying a Verilog task within the task library; and

code for translating the Verilog task into a C++ function; and

if the Verilog file comprises a main driver:

code for inserting in the Verilog file at least one C++ interface header.

22. The machine-readable storage according to claim 20, further comprising code for removing # delay statements from the Verilog file.

23. The machine-readable storage according to claim 20, further comprising code for translating `ifdef statements in the Verilog file.

24. The machine-readable storage according to claim 20, further comprising code for removing ` symbols from the Verilog file.

25. The machine-readable storage according to claim 20, further comprising code for converting a begin keyword in the Verilog file to a "{" symbol.

26. The machine-readable storage according to claim 20, further comprising code for converting an end keyword in the Verilog file to a "}" symbol.

27. The machine-readable storage according to claim 20, further comprising code for converting at least one register definition in the Verilog file into at least one C++ definition.

28. The machine-readable storage according to claim 20, further comprising code for performing at least one combinatorial assignment in the Verilog file.

29. The machine-readable storage according to claim 20, further comprising code for converting at least one event in the Verilog file into at least one C++ event.

30. The machine-readable storage according to claim 20, further comprising code for converting at least one Verilog switch in the Verilog file into at least one C++ switch.

31. The machine-readable storage according to claim 20, further comprising code for converting at least one Verilog concat expressions in the Verilog file into at least one C++ concat expressions.

32. The machine-readable storage according to claim 20, further comprising code for converting at least one Verilog parameter in the Verilog file into at least one C++ #define.

33. The machine-readable storage according to claim 20, further comprising code for converting at least one Verilog const in the Verilog file into at least one C++ const.

34. The machine-readable storage according to claim 20, further comprising code for converting at least one Verilog bit access macro in the Verilog file into at least one C++ functional equivalent.